STATE OF ALABAMA

Information Technology Guideline

Guideline 660-01G4: Error Handling

1. INTRODUCTION:

Improper error handling in an application can lead to an application failure, or possibly result in the application entering an insecure state. Improper error handling is usually not directly attackable; however, during a failure attackers must be prevented from gaining unauthorized access to privileged objects that are normally inaccessible. If upon failing, the system reveals sensitive information about the failure to potential attackers then it's possible for the attacker to cause a Denial of Service (DoS) or possibly exploit the insecure state of the application.

2. OBJECTIVE:

Ensure State of Alabama applications are not subject to error handling vulnerabilities.

3. SCOPE:

This guide may be used for both in-house application development and to assist in the evaluation of the security of third-party applications. The guidance provided is not specific to any one platform, programming language or application type. Some portions of this guide may not apply to all applications.

4. GUIDELINES:

4.1 ERROR HANDLING VULNERABILITY

When applications fail they may go into a non-secure state unless errors are properly handled. Though the most likely effect of improper error handling is a self-inflicted DoS, an attacker could potentially exploit an error handling vulnerability by causing the right kind of failure to happen and bypassing the failed security controls to gain access to the system.

The following may lead to potential error handling issues:

- Failure to check return codes or exceptions
- Improper checking of return codes or exceptions
- Improper handling of return codes or exceptions

The primary way to detect error handling vulnerabilities is to perform code reviews. If a code review cannot be performed, it may be possible to test some error conditions by specifying invalid filenames and using different accounts to run the application. These tests may give indications of vulnerability, but they are not comprehensive.

To minimize error handling vulnerabilities, ensure return code and exception handling is properly and thoroughly implemented throughout the application.

4.2 INFORMATION DISCLOSURE VULNERABILITY

Information disclosure vulnerabilities result from the disclosure of information about the application or application components that may be useful to or the target of a malicious attack. The information itself may be the target of an attacker, or the information could provide an attacker with data needed to compromise the application or system. Information disclosure vulnerabilities are most often the result of programming errors, insufficient authentication, poor error handling or inadequate data protection.

The following may indicate the presence of information disclosure vulnerabilities in an application:

- Information about the operating environment displayed to a user
- Output not marked at the appropriate protection level
- Data access requests not subject to permission checks
- Error messages revealing information to the users through their wording or timing
- Application responses revealing internal information to the user through their wording or timing

The primary method of identifying information disclosure vulnerabilities is to perform a code review. In addition to a code review the application may be tested by:

- Inducing errors in the program to verify the contents of error messages
- Attempting variations of invalid input combinations to determine if the response contents or timing reveal information about the system (For example, is the response time or error message different for an invalid username/invalid password and a valid username/invalid password combination? If so this would allow an attacker to find valid usernames.)
- Attempting to access data the user should not be able to access

In order to minimize Information Disclosure vulnerabilities:

- Ensure a permission structure is in place to enforce access control of the data
- Display generic error messages to end users; provide only enough information so the user knows the request failed
- Ensure application responses do not divulge unneeded details or sensitive information
- Log specific error information details to a secure log file

4.3 DESIGN AND CODING PRINCIPLES

4.3.1 Secure Failure

Applications should perform checks on the validity of data, user permissions, and resource existence before performing a function. Secure failure means if a check fails for any reason, the application code should provide exception handling leaving the application in a secure state. Failing to a secure state means the application has not disclosed any data that would not be disclosed ordinarily and that the data still cannot be tampered with. The principle of secure failure design is intended to account for all possible exceptions that could leave the application in a vulnerable state.

4.3.2 Default to Deny

This principle requires that the default access to an object is none. Whenever access, privileges, or some other security-related attribute is not explicitly allowed, it should be denied. A design or implementation error in an object that requires explicit permission (allow by exception) tends to fail by refusing permission, a safe situation since it will be quickly detected. On the other hand, a design or implementation error in an object that explicitly denies access tends to fail by allowing access, a failure that may go unnoticed.

4.3.3 Graceful Termination

Application processes that receive invalid input should request the external user or process to reinsert the data. If the reinserted data is still invalid, the application should gracefully terminate the user process with an error message to the user indicating that the process is terminating as a result of an input error.

4.3.4 On Failure Undo Changes

Before it shuts down, the application should reverse any changes in its operating mode or state that occurred during execution and return to its normal mode and state of operation.

5. DEFINITIONS:

6. ADDITIONAL INFORMATION:

6.1 POLICY

Information Technology Policy 660-01: Application Security

6.2 RELATED DOCUMENTS

Information Technology Guideline 660-01G2: Input Validation

Signed by Art Bess, Assistant Director

7. DOCUMENT HISTORY:

Version	Release Date	Comments
Original	7/14/2008	